Date: October 2010

TACSIT Controller Interface for CMS Systems

Version 1.0 - Beta 1

OMG Document Number: dtc/2010-10-11
Standard document URL: http://www.omg.org/spec/TACSIT/1.0
Associated Schema Files*: http://www.omg.org/spec/TACSIT/20100801
http://www.omg.org/spec/TACSIT/20100802
http://www.omg.org/spec/TACSIT/20100803

---

* original file(s): c4i/2010-08-02 (UML PSM), c4i/2010-08-03 (Java PSM), c4i/2010-08-04 (C++ PSM)

This OMG document replaces the submission document (c4i/2010-08-01, Alpha). It is an OMG Adopted Beta Specification and is currently in the finalization phase. Comments on the content of this document are welcome, and should be directed to *issues@omg.org* by February 22, 2010.

You may view the pending issues for this specification from the OMG revision issues web page *http://www.omg.org/issues/*.

The FTF Recommendation and Report for this specification will be published on September 24, 2010. If you are reading this after that date, please download the available specification from the OMG Specifications Catalog.

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

LICENSES

The companies listed above have granted to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

GENERAL USE RESTRICTIONS

Any unauthorized use of this specification may violate copyright laws, trademark laws, and communications regulations and statutes. This document contains information which is protected by copyright. All Rights Reserved. No part of this work covered

DISCLAIMER OF WARRANTY

RESTRICTED RIGHTS LEGEND

TRADEMARKS

COMPLIANCE

OMG's Issue Reporting Procedure

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page *http://www.omg.org*, under Documents, Report a Bug/Issue (*http://www.omg.org/technology/agreement.htm*).

Table of Contents

1 Scope 1

2 Conformance Criteria 2

3 Normative References 2

4 Terms and Definitions 3

4.1 General Definitions 3
4.2 Terms specific to this submission 4

5 Acronyms and Abbreviations 5

6 Additional Information 5

6.1 Changes or extensions to OMG specifications 5
6.2 Submitters 5

7 Platform Independent Model (PIM) 7

7.1 Package org.omg.tacsit.controller 7
7.1.1 Class / Interface Specifications 16
7.1.1.1 Interface Entity 16
7.1.1.2 Interface EntityType 18
7.1.1.3 getTypeName 19
7.1.1.4 Interface Projection 19
7.1.1.5 Class SelectionEvent 20
7.1.1.6 Interface SelectionListener 21
7.1.1.7 Interface SelectionManager 22
7.1.1.8 Interface TACSITController 25
7.1.1.9 Class ViewEyeProperties 27
7.1.1.10 Interface Viewport 28
7.1.1.11 ClassViewportChangeEvent 31
7.1.1.12 Interface ViewportChangeListener 32
7.1.1.13 Interface ViewportManager 33
7.1.1.14 Interface ViewportManagerEvent 35
7.1.1.15 InterfaceViewportManagerListener 36
7.1.1.16 EnumerationSelectionMethodology 37
7.1.1.17 EnumerationSelectionType 37
7.2 Package org.omg.tacsit.query 38
7.2.1 Class / Interface Specifications 38
7.2.1.1 Interface ContainmentQuery 38
7.2.1.2 Interface EntityQuery 39
7.2.1.3 Interface EntityTypeQuery 39
7.2.1.4 Interface GeometryQuery 40
7.2.1.5 Interface IntersectionQuery 41
7.2.1.6 InterfaceQueryManager 41

7.3 Package org.omg.tacsit.geometry 42
7.3.1 Class / Interface Specifications 42
7.3.1.1 Interface Geometry 42
7.3.1.2 Interface GeodeticPosition 43
7.3.1.3 Class ScreenPosition 44
7.3.1.4 Interface Centered 45
7.3.1.5 Interface Circle 46
7.3.1.6 Rectangle 47
8 Java Platform Specific Model 49
9 C++ Platform Specific Model 51

Preface

About the Object Management Group

OMG

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable, and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies, and academia.

OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG's specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG's specifications include: UML® (Unified Modeling Language™); CORBA® (Common Object Request Broker Architecture); CWM™ (Common Warehouse Metamodel); and industry-specific standards for dozens of vertical markets.

More information on the OMG is available at *http://www.omg.org/*.

OMG Specifications

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. A Specifications Catalog is available from the OMG website at:

http://www.omg.org/technology/documents/spec_catalog.htm

Specifications within the Catalog are organized by the following categories:

OMG Modeling Specifications

- UML
- MOF
- XMI
- CWM
- Profile specifications

OMG Middleware Specifications

- CORBA/IIOP
- IDL/Language Mappings
- Specialized CORBA specifications
- CORBA Component Model (CCM)

Platform Specific Model and Interface Specifications

- CORBAservices
- CORBAfacilities
- OMG Domain specifications
- OMG Embedded Intelligence specifications
- OMG Security specifications

All of OMG's formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format, may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. at:

OMG Headquarters
140 Kendrick Street
Building A, Suite 300
Needham, MA 02494
USA
Tel: +1-781-444-0404
Fax: +1-781-444-0320
Email: *pubs@omg.org*

Certain OMG specifications are also available as ISO standards. Please consult *http://www.iso.org*

Typographical Conventions

The type styles shown below are used in this document to distinguish programming statements from ordinary English. However, these conventions are not used in tables or section headings where no distinction is necessary.

Times/Times New Roman - 10 pt.: Standard body text

Helvetica/Arial - 10 pt. Bold: OMG Interface Definition Language (OMG IDL) and syntax elements.

`Courier - 10 pt. Bold:` Programming language elements.

Helvetica/Arial - 10 pt: Exceptions

Note – Terms that appear in *italics* are defined in the glossary. Italic text also represents the name of a document, specification, or other publication.

Issues

The reader is encouraged to report any technical or editing issues/problems with this specification to *http://www.omg.org/technology/agreement.htm*.

1 Scope

The domain of Combat Management Systems is characterized by a huge variety of underlying computing platforms, with different and often incompatible means of providing interactive displays. Standards-based services are essential for interoperable and open systems.

There is fairly broad agreement of what is considered the TACSIT software component of a tactical / strategic display system. The TACSIT component is the software that provides users awareness of entities in the operational space relative to a certain geo-spatial context. The TACSIT, by its nature, displays entities called tracks in their proper geographic location overlaid on a visual representation of a map while including additional annotations and decision aides to support the operator. The TACSIT is distinct from other display applications that work around, or in conjunction with it. The figure below provides an example of the TACSIT software component.

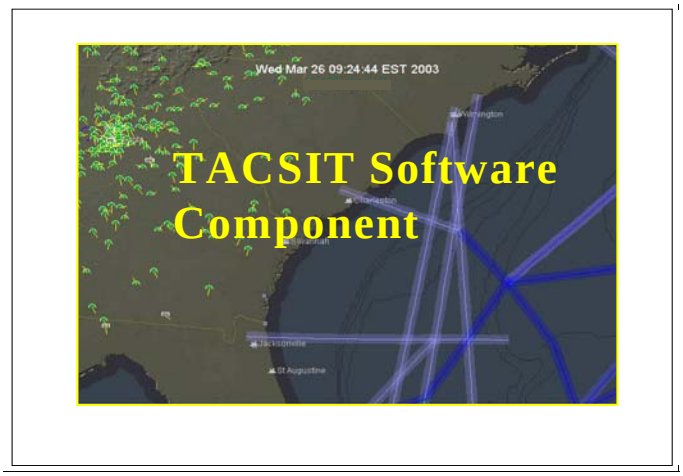


Figure 1.1 - Example of a TACSIT software component

There are many capabilities and services necessary to successfully implement a TACSIT software component.

Within this broader context of TACSIT services, there are several aspects to be considered such as Controllers, Maps, Tracks, Geometric, and Configuration. This specification is targeted at the Controller services necessary for TACSIT software component manipulation only. Controller Services provide a means to interact with the TACSIT software component and receive TACSIT notifications. Some general examples of Controller Services capability include such functionalities as range scaling, setting the area center/offset, setting view rotation, selecting objects, registering for events, and receiving cursor location updates.

This document specifies a standard for TACSIT Controller software services in CMS systems, consisting of a standard means of interacting with TACSIT software components.

The context of this TACSIT Controller service is defined in the following figure and associated text:

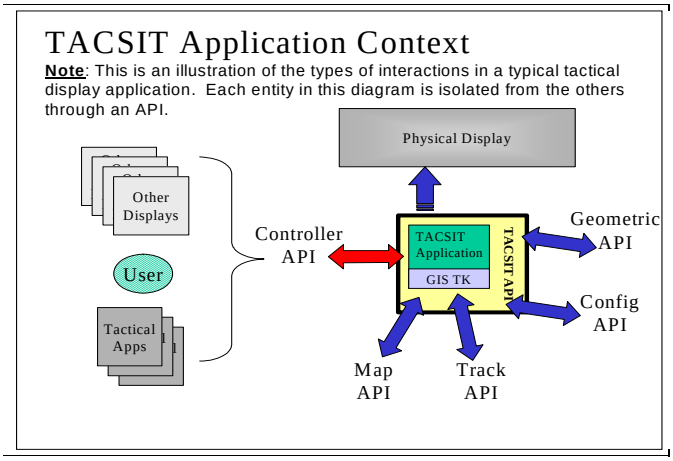


Figure 1.2 - TACSIT Application Context

This specification focuses on the interfaces necessary for software applications to interact with the TACSIT application, to modify the state of the TACSIT configuration, and to listen for relevant state changes that occur within the TACSIT (often as a result of user interaction).

2 Conformance Criteria

Implementations of this standard are considerd to be in conformance if they match one or more of the language level PSMs specified. The implementation must indicate the langage PSM(s) that they match in their statement of conformance.

3 Normative References

The team considered the OGC “OpenGIS® Implementation Specification for Geographic information - Simple feature access - Part 1: Common architecture” (Doc: OGC 06-103r3; Version: 1.2.0; Date: 2006-10-05) in that:

- This OMG specification does not conflict with the OGC specification and uses the same names for types and methods where possible.
- This OMG Specification does not require a Geometry to completely implement the OGC specification.
- The OGC specification is considerably more involved and complex than the requirements for this standard therefore we have specified a simple Geometry class for this specification.

4 Terms and Definitions

4.1 General Definitions

Architecture Board (AB) - The OMG plenary that is responsible for ensuring the technical merit and MDA-compliance of RFPs and their submissions.

Board of Directors (BoD) - The OMG body that is responsible for adopting technology.

Common Object Request Broker Architecture (CORBA) - An OMG distributed computing platform specification that is independent of implementation languages.

Common Warehouse Metamodel (CWM) - An OMG specification for data repository integration.

CORBA Component Model (CCM) - An OMG specification for an implementation language independent distributed component model.

Interface Definition Language (IDL) - An OMG and ISO standard language for specifying interfaces and associated data structures.

Letter of Intent (LOI) - A letter submitted to the OMG BoD's Business Committee signed by an officer of an organization signifying its intent to respond to the RFP and confirming the organization's willingness to comply with OMG's terms and conditions, and commercial availability requirements.

Mapping - Specification of a mechanism for transforming the elements of a model conforming to a particular metamodel into elements of another model that conforms to another (possibly the same) metamodel.

Metadata - Data that represents models. For example, a UML model; a CORBA object model expressed in IDL; and a relational database schema expressed using CWM.

Metamodel - A model of models.

Meta Object Facility (MOF) - An OMG standard, closely related to UML, that enables metadata management and language definition.

Model - A formal specification of the function, structure and/or behavior of an application or system.

Model Driven Architecture (MDA) - An approach to IT system specification that separates the specification of functionality from the specification of the implementation of that functionality on a specific technology platform.

Normative - Provisions that one must conform to in order to claim compliance with the standard. (as opposed to non-normative or informative which is explanatory material that is included in order to assist in understanding the standard and does not contain any provisions that must be conformed to in order to claim compliance).

Normative Reference - References that contain provisions that one must conform to in order to claim compliance with the standard that contains said normative reference.

Platform - A set of subsystems/technologies that provide a coherent set of functionality through interfaces and specified usage patterns that any subsystem that depends on the platform can use without concern for the details of how the functionality provided by the platform is implemented.

Platform Independent Model (PIM) - A model of a subsystem that contains no information specific to the platform, or the technology that is used to realize it.

Platform Specific Model (PSM) - A model of a subsystem that includes information about the specific technology that is used in the realization of it on a specific platform, and hence possibly contains elements that are specific to the platform.

Request for Information (RFI) - A general request to industry, academia, and any other interested parties to submit information about a particular technology area to one of the OMG's Technology Committee subgroups.

Request for Proposal (RFP) - A document requesting OMG members to submit proposals to the OMG's Technology Committee. Such proposals must be received by a certain deadline and are evaluated by the issuing task force.

Task Force (TF) - The OMG Technology Committee subgroup responsible for issuing a RFP and evaluating submission(s).

Technology Committee (TC) - The body responsible for recommending technologies for adoption to the BoD. There are two TCs in OMG - Platform TC (PTC), that focuses on IT and modeling infrastructure related standards; and Domain TC (DTC), that focus on domain specific standards.

Unified Modeling Language (UML) - An OMG standard language for specifying the structure and behavior of systems. The standard defines an abstract syntax and a graphical concrete syntax.

UML Profile - A standardized set of extensions and constraints that tailors UML to particular use.

XML Metadata Interchange (XMI) - An OMG standard that facilitates interchange of models via XML documents.

4.2 Terms specific to this submission

The RFP prompting this response defined the following set of standard terminology which will henceforth be used within this document:

- TACSIT - A Tactical Situation Display software component that provides a display of relevant tactical information over and in conjunction with the geographic context of the information.
- Track - A spatial object that is managed within the CMS, such as local radar contacts, radar contacts provided via external messages, tactical data points, waypoints, etc.
- Viewport - The area of visible geography displayed in the TACSIT window.
- Hook Event - The selection of a track on the TACSIT. The words "selected" and "hooked" are used interchangeably within this document.
- Primary Hook - The predominant hooked track on the TACSIT.
- Secondary Hook - An alternate or subordinate hooked track on the TACSIT.
- Pre-Hook -The case where the pointer is moved over a track, but no selection / hook action has been initiated by the operator or the CMS. This condition presumes a particular human interface that includes the use of a pointing device.
- CMS - Combat Management System - The software systems and services used in providing command and control services to a system.
- C4I - A term used in military and command situations that is an abbreviation for Command, Control, Computers, Communications, and Inteligence.

5 Acronyms and Abbreviations

Table 5.1 - Acronyms and Abbreviations

CMS	(Naval) Combat Management System
CORBA	Common Object Request Broker Architecture
TACSIT	Tactical Situation (Display)
HTTP	HyperText Transfer Protocol
OMG	Object Management Group
RFP	Request For Proposal
UML	Unified Modelling Language
XML	eXtensible Mark-up Language

6 Additional Information

6.1 Changes or extensions to OMG specifications

No changes to UML 2.0 or other OMG specifications are required.

6.2 Submitters

- BAE Systems plc
- LUCIAD
- SimVentions Inc

7 Platform Independent Model (PIM)

The PIM consists of the following logical packages:

- TACSIT Controller Interface (org.omg.tacsit.controller)
- TACSIT Query (org.omg.tacsit.query)
- TACSIT Geometry (org.omg.tacsit.geometry)

7.1 Package org.omg.tacsit.controller

The TACSIT Controller package contains most of the major interfaces of this specification. They are categorized in three logical groups: Top Level, Selection, and Viewport.

Top Level Interfaces and Relationships

The top level interface for this specification is the TACSITController interface. It provides the primary façade for working with the services herein. The following class diagram shows the relationship between the TACSITController façade and the various manager interfaces used to implement the functionality of this specification.

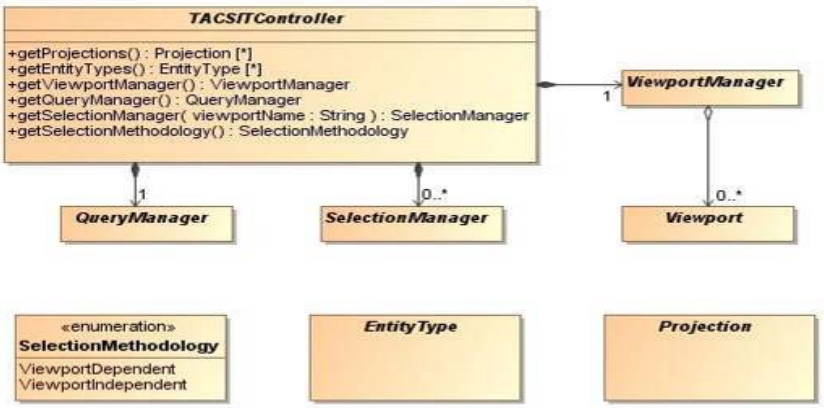


Figure 7.1 - Top Level Interfaces

For a detailed description of the these classes, see the definitions outlined in section 6.3.2.

Selection Interfaces and Relationships

The selection functionality of this stanard is contained in the org.omg.tacsit.controller package. They provide the interfaces for working with the selected item in the TACSIT. The following class diagram shows the relationship between the various interfaces used to implement the functionality of this portion of the specification.

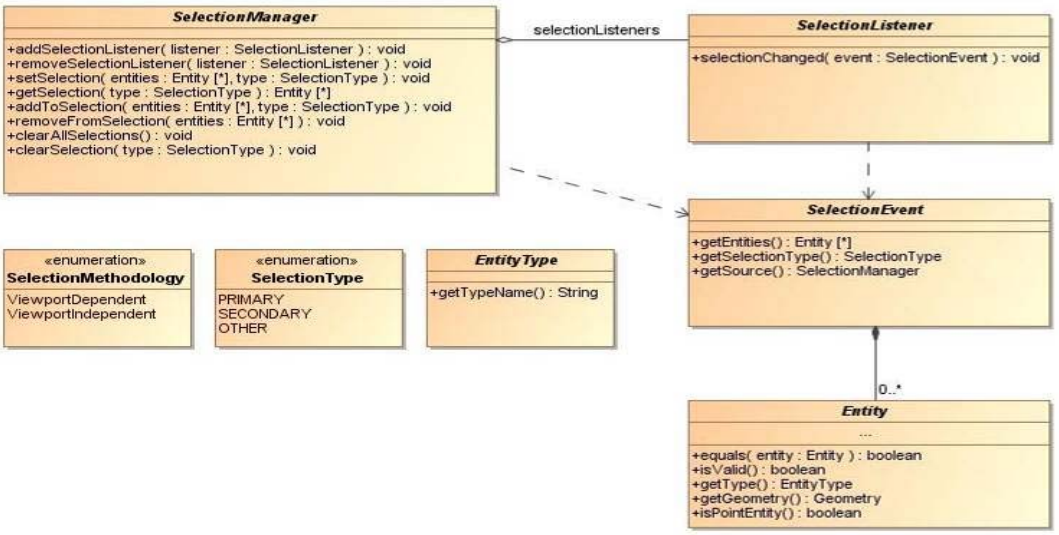


Figure 7.2 - Selection Interfaces

These interfaces are designed to support the common selection requirements involved in a TACSIT system.

These interfaces support the ability to get and set the currently selected item (primary and secondary) via API call. There are two “reserved” named selections (hooks), primary and secondary, corresponding to SelectionTypes PRIMARY and SECONDARY. Using these interfaces provides considerable flexibility in managing the selected items and allows for the following functionality.

- Select by entity / track number (TN), with the ability to specify the type or category of TN (as many CMS support multiple categories of TN per track such as System TN, Link TN, etc.).
- Select tracks within a geographic geometry such as a bounding box.
- Select tracks at a geographic point where the parameters allow for selection of the closest track(s) within a radius of the specified point.
- Select tracks with particular attribute(s) such as (Surface, Hostile, Frigate).

The following sequence patterns demonstrate how to perform these functions. These patterns support working with single entities and groups of entities.

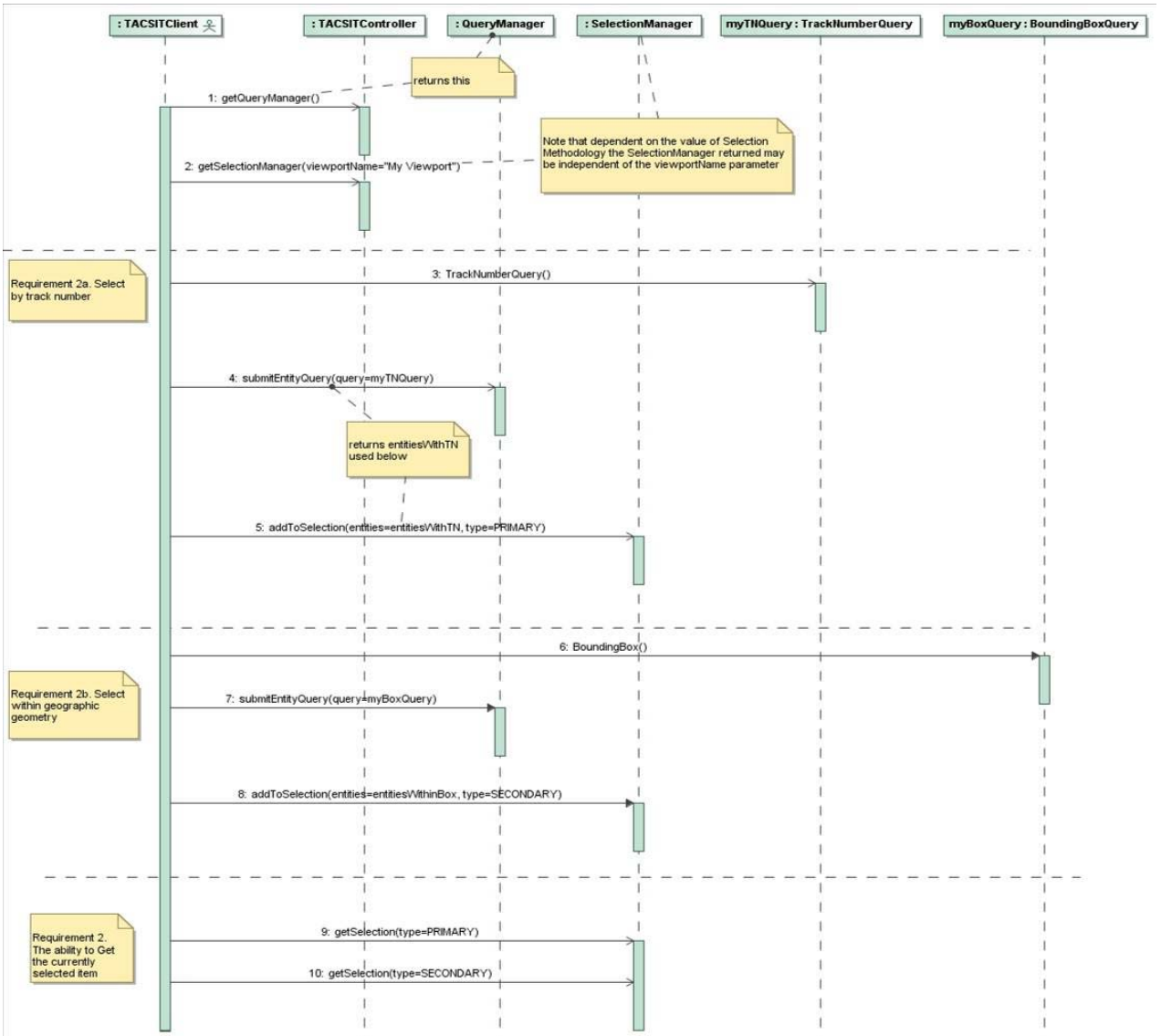


Figure 7.3 -

To remove objects (including tracks) from the current selection follow the sequence diagram below. You may remove all objects, a single object, multiple objects, and all objects with a particular attribute.

- A method to accomplish this is shown in the sequence diagram below.

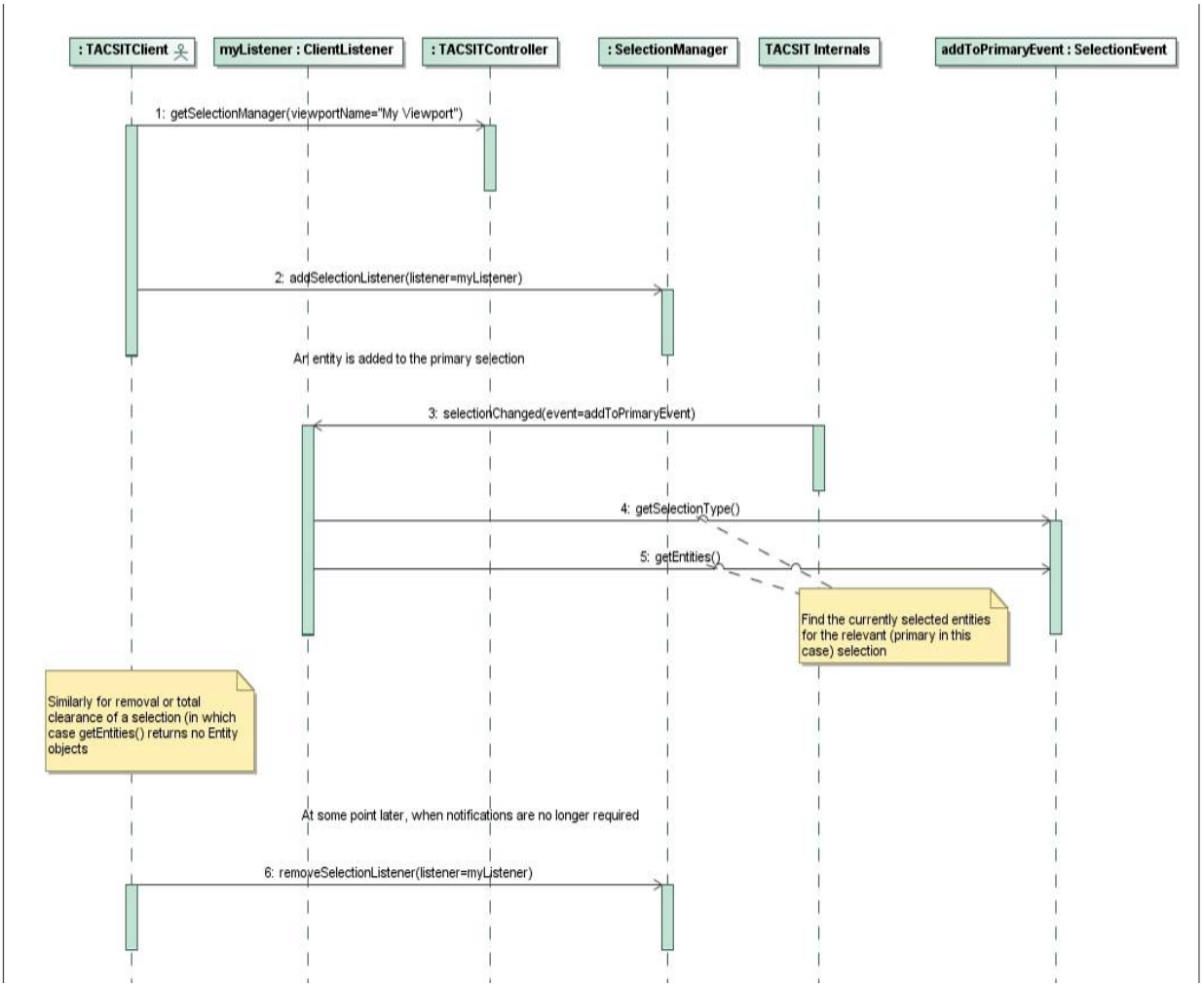
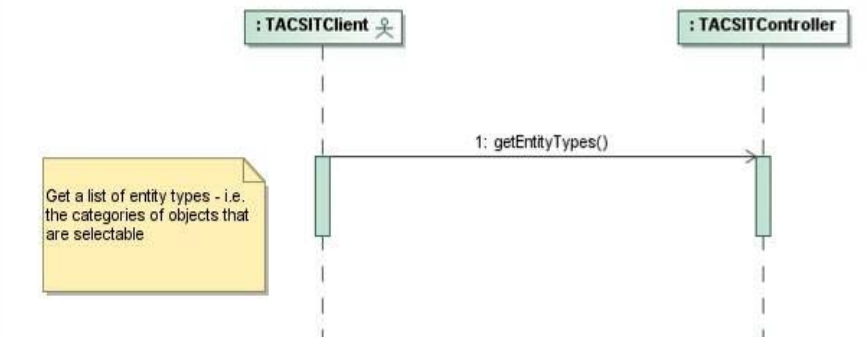


Figure 7.4 -

These interfaces allow you to get and set the category(ies) of selectable items and specify the categories of objects that are selectable (e.g. vehicular, non-vehicular tracks, countries, overlays, tactical points, reference points, waypoints, etc.).



Viewport Interfaces and Relationships

These interfaces are designed to support the common viewport requirements involved in a TACSIT system.

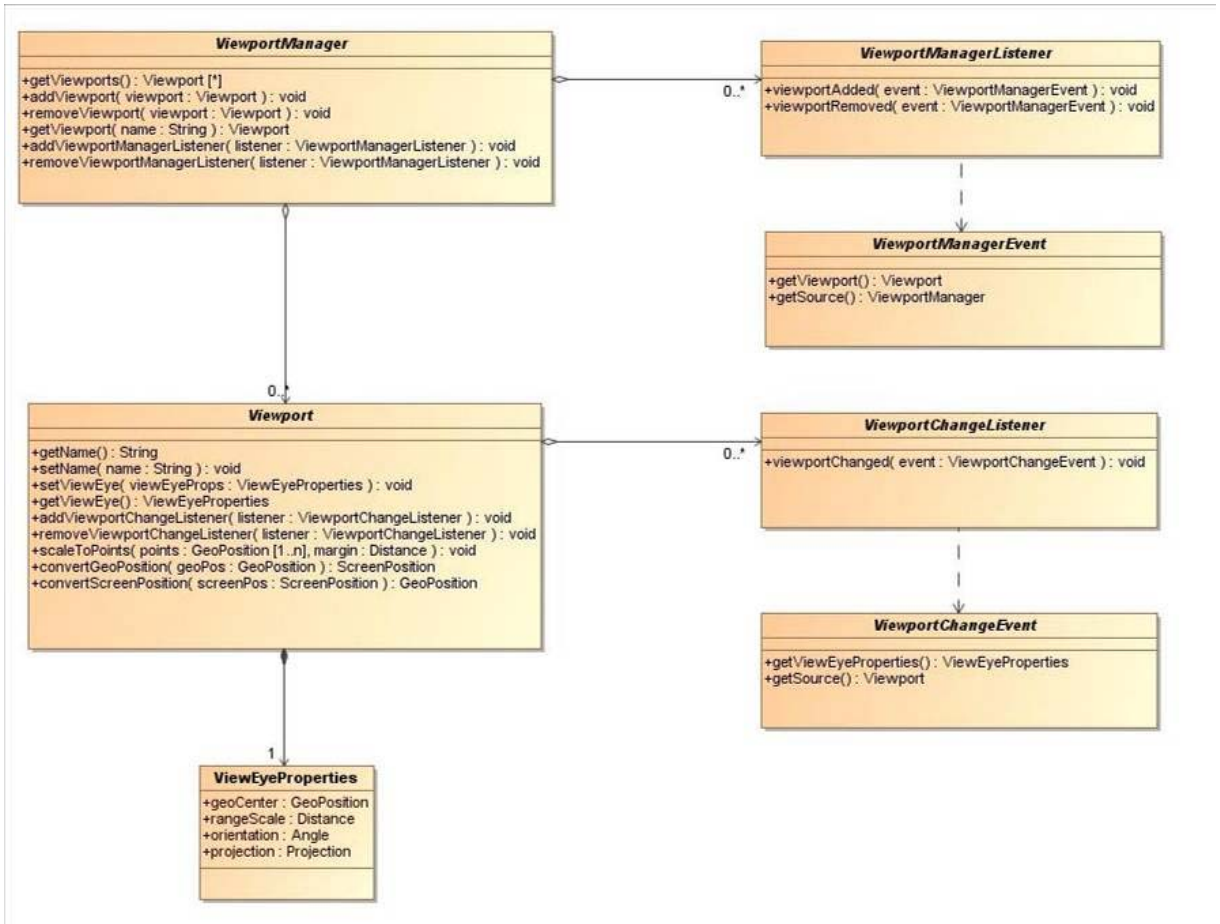
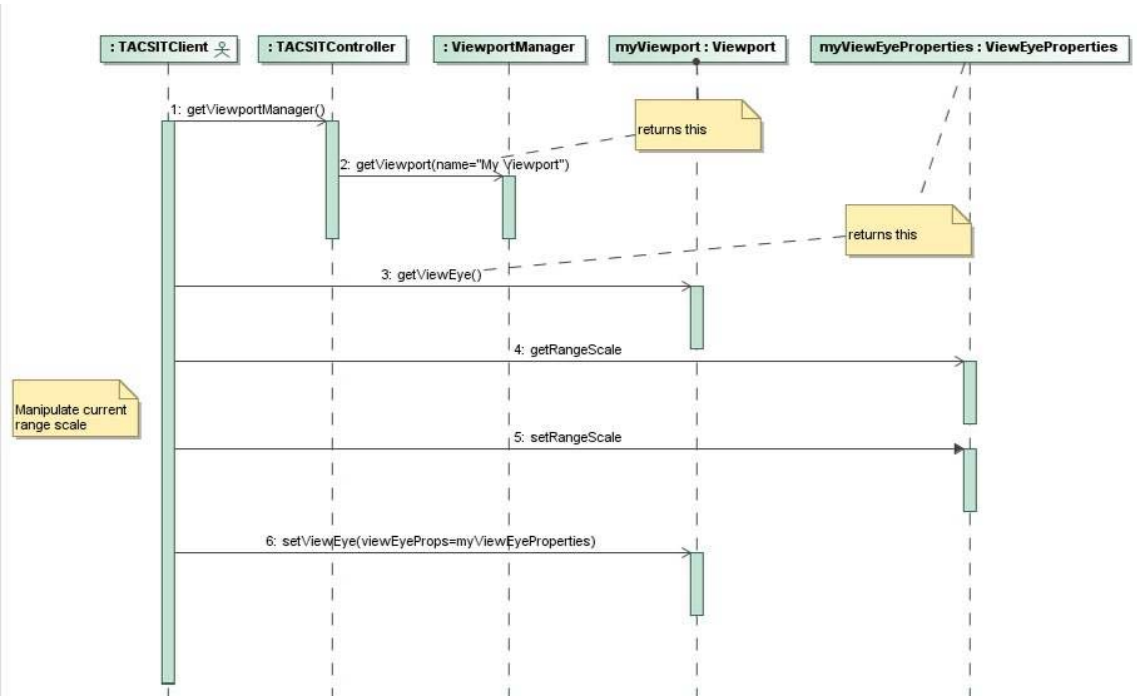


Figure 7.5 - Viewport Interfaces

These interfaces are designed to support the common viewport related requirements involved in a TACSIT system. These interfaces support the ability to manage the current view of the TACSIT. The following section describes the functionality provided.

To get and set the current range scale (distance from center to nearest edge).



Get and set the viewport range boundaries (within the TACSIT window) by specifying two geographic points (Point 1, Point 2) and a distance buffer outside 1 and 2.

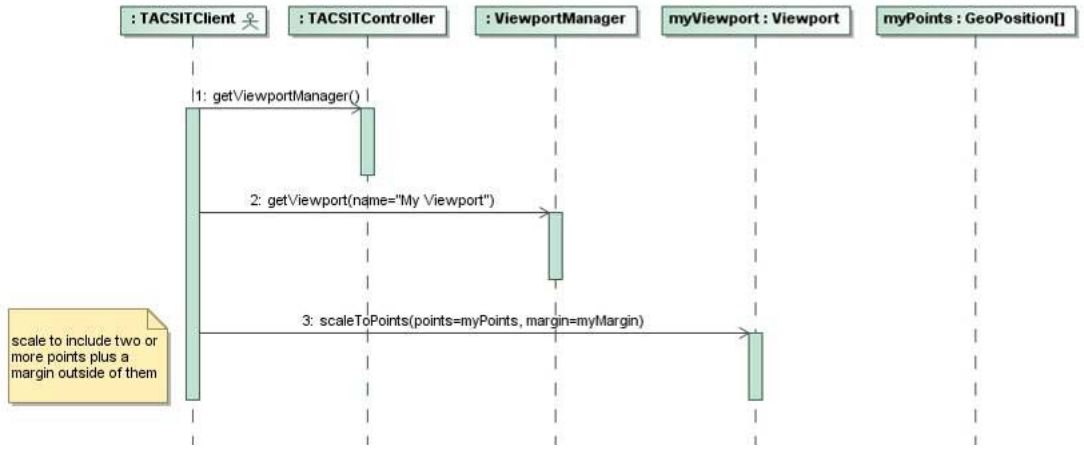


Figure 7.6 -

Get and set the current display center specificed as geographic position.

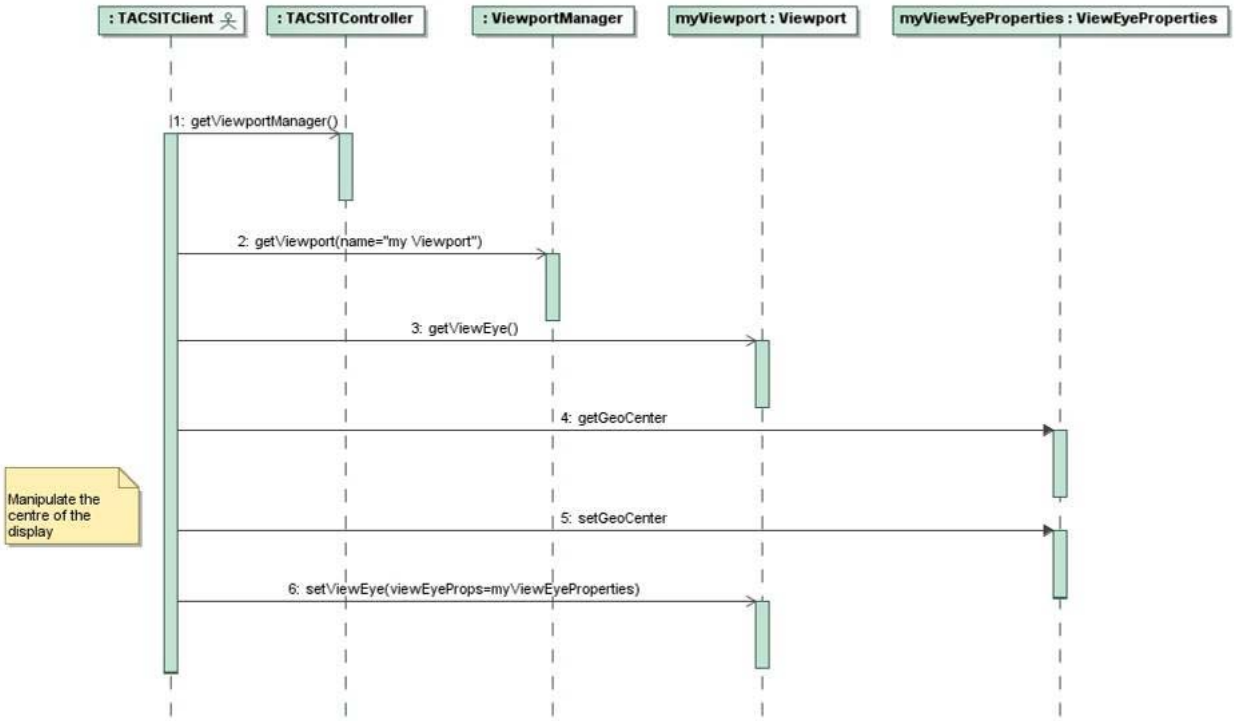


Figure 7.7 -

You can register listeners for changes in the TACSIT viewport extents including range scale, center.

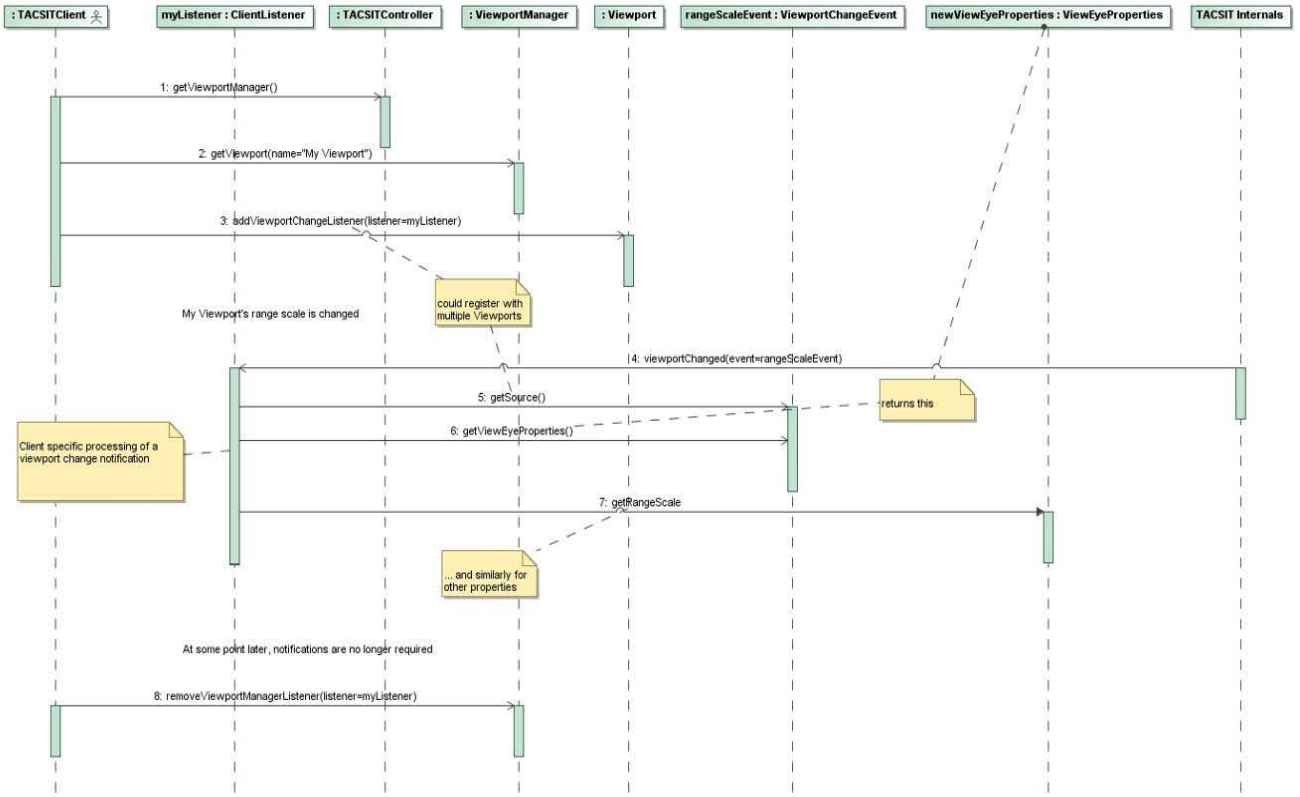


Figure 7.8 -

Set the center of the TACSIT display to an object. If your system requires the center to be consistently updated with the movement with a particular entity, you must periodically (based on your own timing and performance requirements) call these methods as required.

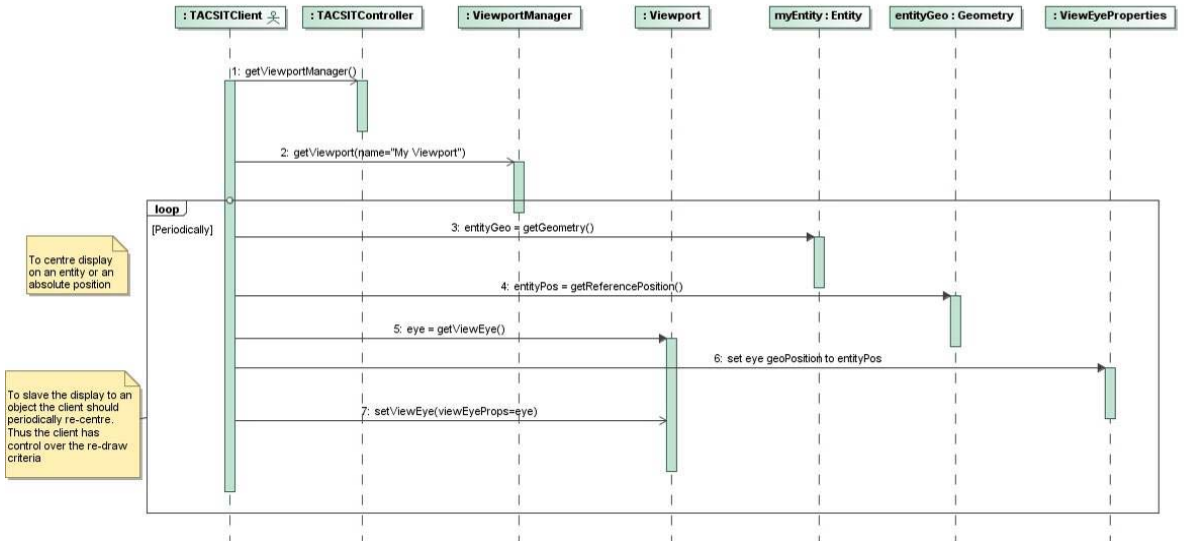


Figure 7.9 -

Get and set the geographic projection used for a particular Viewport. To get a list of available projections call the method in the TACSITController class. This provides a simple list of objects that give the names of the projections supported by the service. You may then use this object to set the projection desired in your viewport.

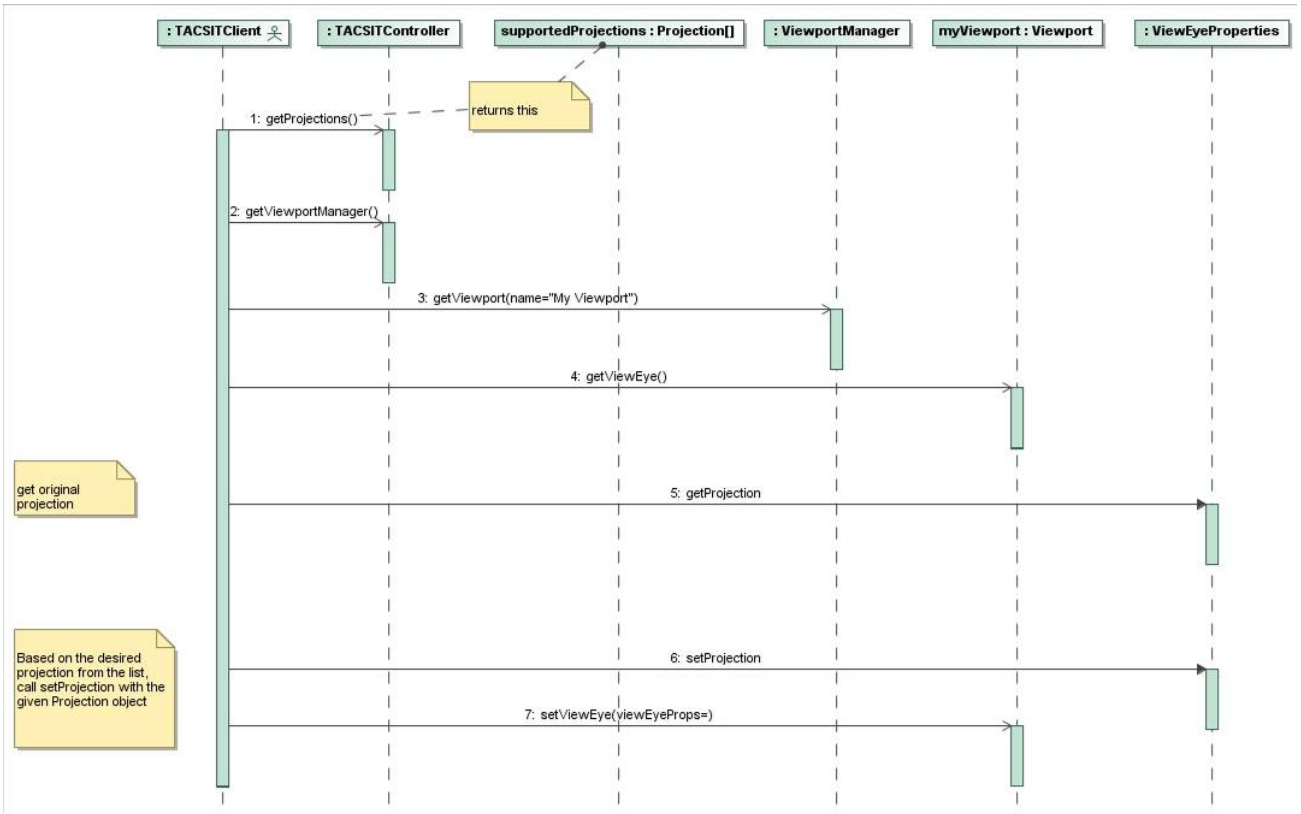


Figure 7.10 -

7.1.1 Class / Interface Specifications

The following provides a detailed description of the interface classes in the query package.

7.1.1.1 Interface Entity

The Entity class is an abstraction of the tactical objects that can be displayed on a Tactical Situation display (TacSit).

The Entity interface specifies the basic methods of interaction between the TACSIT system and the custom clients. Each application uses their own specific objects for Tracks, items of interest, etc and they implement the Entity interface. While not specified in this specification, it is assumed that these types of Entity objects are the ones “added” by the application to the TACSIT to be visualized. Having this Entity interface allows both the TACSIT Controller implementation and the user-system to share tracks and other “real-world” objects.

Entities are modeled such that it is possible for clients to establish their own specific type and geometry and hence proceed to interact with them in an application specific manner.

Entities need to be able to indicate if they are still valid: even though they may have been part of a selection at a certain point in time, by the time that the entities of that selection are accessed the entities may not exist any more in reality.

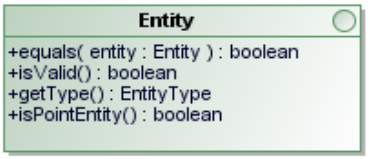


Figure 7.11 - Entity

Name	Entity
Qualified Name	org::omg::tacsit::controller::Entity
Visibility	public
Base Classifier	

7.1.1.1.1 equals

Returns true if the given Entity corresponds to this Entity (i.e., the Tactical Objects are the same).

Type	boolean
Visibility	public
Is Abstract	false
Parameter	in entity : Entity

7.1.1.1.2 getType

Returns the type of the Entity. This corresponds to the application specific class of the tactical object that has been added to the TacSit.

Type	EntityType
Visibility	public
Is Abstract	false
Parameter	

7.1.1.1.3 isPointEntity

This is a convenience method for indicating whether the entity is to be handled as though it is a “single” point rather than a complex geometry. This is useful for containment and other queries.

Type	boolean
Visibility	public
Is Abstract	false
Parameter	

Type	boolean
Visibility	public
Is Abstract	false
Parameter	

Type	boolean
Visibility	public
Is Abstract	false
Parameter	

7.1.1.1.4 isValid

Returns true if this instance is valid.

Type	boolean
Visibility	public
Is Abstract	false
Parameter	

7.1.1.2 Interface EntityType

An EntityType represents the type of an Entity that is handled by the Tacsit.

An EntityType is an implicit grouping of Entities according to their characteristics in reality (e.g., Track, Airfield, Country, etc.)

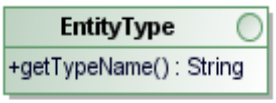


Figure 7.12 - Entity Type

Name	EntityType
Qualified Name	org::omg::tacsit::controller::EntityType
Visibility	public
Base Classifier	

7.1.1.3 getTypeName

Returns the display name of this EntityType.

Type	String
Visibility	public
Is Abstract	false
Parameter	

7.1.1.4 Interface Projection

A projection is a method of mapping the curvi-linear 3D geometry of the Earth's surface to a 2D TACSIT Viewport (i.e., a Map Projection).

For the purpose of the Tacsit Controller interface projections are only needed to refer to by name.

Each "Toolkit" is free to provide whatever projections it supports. This interface provides the users access to the individually supported projections without having to specify or understand the details of how they are implemented. For projection details (such as projection parameters) toolkit specific provisions are foreseen.

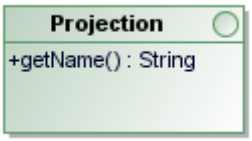


Figure 7.13 - Projection

Name	Projection
Qualified Name	org::omg::tacsit::controller::Projection
Visibility	public
Base Classifier	

7.1.1.4.1 getName

Returns the name of this Projection.

Type	String
Visibility	public
Is Abstract	false
Parameter	

7.1.1.5 Class SelectionEvent

SelectionEvent is the type of event that is passed to a SelectionListener each time there is a change in one of the selections of a SelectionManager.

Figure 7.14 - SelectionEvent

Name	SelectionEvent
Qualified Name	org::omg::tacsit::controller::SelectionEvent
Visibility	public
Abstract	false
Base Classifier	
Realized Interface	

7.1.1.5.1 getEntities

Returns the list of entities involved in this selection event.

Type	Entity
Visibility	public
Is Abstract	false
Parameter	

7.1.1.5.2 getSelectionType

Returns the type of selection that has changed.

Type	SelectionType
Visibility	public
Is Abstract	false
Parameter	

7.1.1.5.3 getSource

Returns the SelectionManager in which the selection change occurred.

Type	SelectionManager
Visibility	public
Is Abstract	false
Parameter	

7.1.1.5.4 SelectionEvent

A constructor to build and fill the attributes of the SelectionEvent.

Type	Constructor
Visibility	public
Is Abstract	false
Parameter	• in entities : Entity [0..*] • inout source : SelectionManager • inout type : SelectionType

7.1.1.6 Interface SelectionListener

SelectionListener is the type of object that is notified by a SelectionManager where it is registered in case of a change in one of the selections of that SelectionManager.

Figure 7.15 - SelectionListener

Name	SelectionListener
Qualified Name	org::omg::tacsit::controller::SelectionListener
Visibility	public
Base Classifier	

7.1.1.6.1 selectionChanged

This method is called by the SelectionManagers where this SelectionListener is registered in case a selection changed for that SelectionManager. The details of which selection changed and which elements are still in the selection can be obtained through the given SelectionEvent.

Type	void
Visibility	public
Is Abstract	false
Parameter	• in event : SelectionEvent

7.1.1.7 Interface SelectionManager

The selection manager is the entry point for managing the selections of the Tacsit controller.

Multiple selections can be managed using the selection manager. Each selection has a specific selection type. Selection types are predefined. Each selection is managed as a list of the type Entity. Each element can only occur once in the selection and the order of the elements in the selection is not determined.

Managing a selection through the selection manager means that the selection manager allows retrieving and modifying each of the selections. It also allows users to register and unregister selection listeners which will be notified in case of a change in a selection.

The scope of the SelectionManager is determined by the SelectionMethodology of the Tacsit controller.

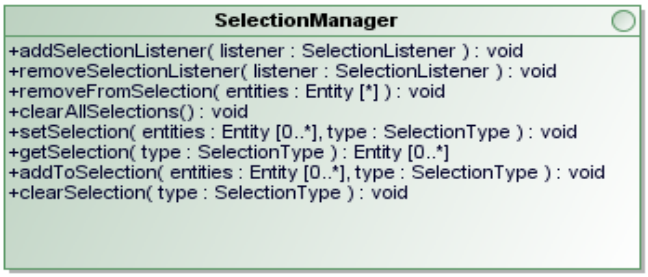


Figure 7.16 - SelectionManager

Name	SelectionManager
Qualified Name	org::omg::tacsit::controller::SelectionManager
Visibility	public
Base Classifier	

7.1.1.7.1 addSelectionListener

Registers the given SelectionListener to this SelectionManager.

After registration the SelectionListener will be notified through its only method each time that a change occurs in one of the selections managed by this SelectionManager. Registering a SelectionListener that was already registered with this SelectionManager therefore does not have any effect.

Type	void
Visibility	public
IsAbstract	false
Parameter	• in listener : SelectionListener

7.1.1.7.2 addToSelection

Adds all elements of the given array of Entities to the selection of the given SelectionType of this SelectionManager. The given EntityList should not be empty.

After calling this method, getSelection (type) contains all elements of the given array of Entities as well as any entities previously selected for that type.

Each Entity can be part of only one selection at a time. Therefore, all entities of the given array of Entities that were part of a selection of another SelectionType than the given type are removed from those selections.

All SelectionListeners registered at the time of the addToSelection are notified by a call of their method selectionChanged.

Type	void
Visibility	public
IsAbstract	false
Parameter	• in entities : Entity [*] • in type : SelectionType

7.1.1.7.3 clearAllSelections

Clears all selections of this SelectionManager.

After calling this method, the selection for any SelectionType contains no Entities.

All SelectionListeners registered at the time of the clearAllSelections are notified by a call of their method selectionChanged.

Type	void
Visibility	public
IsAbstract	false
Parameter	

7.1.1.7.4 clearSelection

Clears the selection of the given SelectionType of this SelectionManager.

Afterwards the selection of the given SelectionType of this SelectionManager does not contain any Entities. All other selections of this SelectionManager are unchanged.

All SelectionListeners registered at the time of the clearSelection are notified by a call of their method selectionChanged.

Type	void
Visibility	public
IsAbstract	false
Parameter	• in type : SelectionType

7.1.1.7.5 getSelection

Returns a list of selected Entities of the given SelectionType of this SelectionManager. Note that editing the returned EntitySet does not have an impact on the selection of the given SelectionType of this SelectionManager.

Type	Entity
Visibility	public
IsAbstract	false
Parameter	• in type : SelectionType

7.1.1.7.6 removeFromSelection

Removes all Entities in the given EntitySet from the selection to which they belong. Afterwards, these entities will not be contained in any selection until added (where contains is based on equals).

All SelectionListeners registered at the time of the removeFromSelection are notified by a call of their method selectionChanged for each SelectionType from which at least one Entity was removed.

Type	Entity
Visibility	public
IsAbstract	false
Parameter	• in entities : Entity [*]

7.1.1.7.7 removeSelectionListener

Unregisters the given SelectionListener from this SelectionManager. Afterwards the SelectionListener will not be notified of changes in the selections of this SelectionManager.

Unregistering a SelectionListener that was not registered therefore does not have any effect.

Type	void
Visibility	public
IsAbstract	false
Parameter	• in listener : SelectionListener

7.1.1.7.8 setSelection

Changes the selection of the given SelectionType to the given EntitySet. The given EntitySet should not be empty (see clearSelection and clearAllSelections). Afterwards the set of Entities of getSelection(type) equals the set of Entities of the given EntitySet.

All SelectionListeners registered at the time of the setSelection are notified by a call of their method selectionChanged.

Type	void
Visibility	public
IsAbstract	false
Parameter	• in entities: Entity [*] • in type : SelectionType

7.1.1.8 Interface TACSITController

This is the top level façade class for the Tactical Situation display (TACSIT) Controller API which enables clients to interact with SelectionManagers, Viewports, and Query TacSit content in terms of the Entity class.

The TacSit Controller API does not provide the mechanism for adding Entities to a TacSit. When a client receives an Entity from a Query, the client of the TacSit controller will interpret the Entity according to interfaces provided by other components than the Tacsit Controller (for instance an interaction with another component, which is known to have added the Entity to the TacSit by some other API.

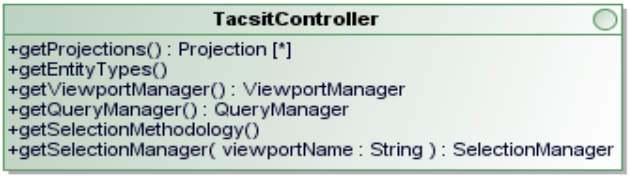


Figure 7.17 - TacsitController

Name	TacsitController
Qualified Name	org::omg::tacsit::controller::TacsitController
Visibility	public
Base Classifier	

7.1.1.8.1 getProjections

Returns the projections that are supported by the TACSIT.

Type	Projection
Visibility	public
IsAbstract	false
Parameter	

7.1.1.8.2 getEntityTypes

Returns the Entity Types that are supported by the TACSIT. This will return a list of all Entity Types currently available for Selection and Query by this TACSITController.

Type	EntityType
Visibility	public
IsAbstract	false
Parameter	

7.1.1.8.3 getQueryManager

Returns the QueryManager for this Tacsit Controller.

Type	QueryManager
Visibility	public
IsAbstract	false
Parameter	

7.1.1.8.4 getSelectionManager

Returns the SelectionManager that handles the selection on the viewport with name viewportName.

Depending on the SelectionMethodology of the Tacsit Controller the returned SelectionManagers for multiple viewports (different named viewports) will be identical or different. If the SelectionMethodology is ViewportDependent then a different SelectionManager is returned for each individual viewport. If SelectionMethodology is ViewportIndependent, then one SelectionManager is used for all viewports, thus making selection “global” within the scope of this TACSITController.

Type	SelectionManager
Visibility	public
IsAbstract	false
Parameter	• in viewportName : String

7.1.1.8.5 getSelectionMethodology

Returns the SelectionMethodology of the Tacsit controller. If the SelectionMethodology is ViewportDependent then each viewport is managed by its own SelectionManager. If SelectionMethodology is ViewportIndependent, then one SelectionManager is used for all viewports, thus making selection “global” within the scope of this TACSITController.

Type	SelectionMethodology
Visibility	public
IsAbstract	false
Parameter	

7.1.1.8.6 getViewportManager

Returns the ViewportManager for this Tacsit Controller.

Type	ViewportManager
Visibility	public
IsAbstract	false
Parameter	

7.1.1.9 Class ViewEyeProperties

ViewEyeProperties groups the attributes of a Viewport that have an impact on the position where Entities are displayed in the Viewport.

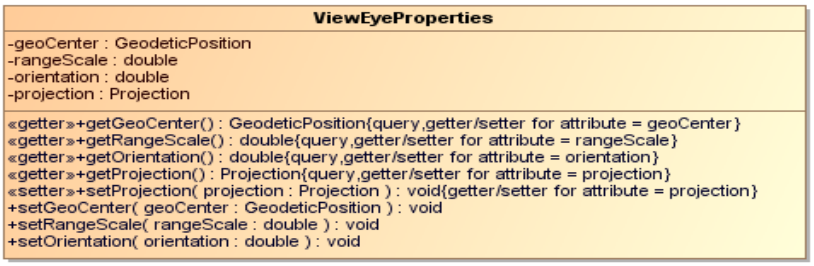


Figure 7.18 - ViewEyeProperties

Name	ViewEyeProperties
Qualified Name	org::omg::tacsit::controller::ViewEyeProperties
Visibility	public
Abstract	false
Base Classifier	
Realized Interface	

7.1.1.9.1 geoCenter

The geographic center of the Viewport. The geoCenter of the Viewport is the Geoposition that is displayed in the center of the Viewport (i.e., on a Viewport of w pixels wide and h pixels high, it is displayed on pixel (w/2, h/2)).

Type	GeoPosition
Default Value	
Visibility	public
Multiplicity	1

7.1.1.9.2 orientation

The orientation of the Viewport.

The orientation is applied on the Viewport as an Angle rotation of the Viewport in clockwise direction around the Viewport's center. See the definition of Angle for precision guidance.

Type	Angle
Default Value	
Visibility	public
Multiplicity	1

7.1.1.9.3 projection

The projection of the Viewport. The projection object here is assumed to be one of the Projections provided by the TACSIT Controller method “getProjections().” This specification does not address any further details of how to handle projections.

Type	Projection
Default Value	
Visibility	public
Multiplicity	

7.1.1.9.4 rangeScale

The rangescale of the Viewport. The rangescale is the Distance from the center of the viewport to the nearest viewport edge.

Type	Distance
Default Value	
Visibility	public
Multiplicity	1

7.1.1.9.5 getters/setters

Each of the the attributes of the ViewEyeProperties class has an associated getter and setter method matching in name and type with the attribute. See the class image above for the signature for these methods.

7.1.1.10 Interface Viewport

The Viewport class enables a TacSit client to interact with the basic properties of a TacSit view. In particular the view's name, where it is centered, how it is scaled, projected and oriented, and the selections it has (through its SelectionManager instance).

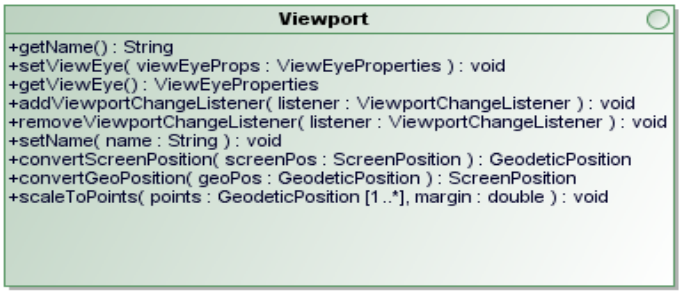


Figure 7.19 - Viewport

Name	Viewport
Qualified Name	org::omg::tacsit::controller::Viewport
Visibility	public
Base Classifier	

7.1.1.10.1 addViewportChangeListener

Registers the given ViewportChangeListener to this Viewport. After registration the ViewportChangeListener will be notified through its only method each time that a viewport is changed. Registering a ViewportChangeListener that was already registered with this Viewport does not have any effect.

Type	void
Visibility	public
IsAbstract	false
Parameter	• in listener : ViewportChangeListener

7.1.1.10.2 convertGeoPosition

Converts the given GeodeticPosition into a ScreenPosition (pixels). Note that the conversion will not be valid if the give position is not projectable to the screen in that case an error condition is returned (specified by the PSM).

Type	ScreenPosition
Visibility	public
IsAbstract	false
Parameter	• in geoPos : GeodeticPosition

7.1.1.10.3 convertScreenPosition

Converts the passed screen position (pixels) into a geo position. Note that the conversion will not be valid if the give screen position is not projectable to the earth model in that case an error condition is returned (specified by the PSM).

Type	GeodeticPosition
Visibility	public
IsAbstract	false
Parameter	• in screenPos : ScreenPosition

7.1.1.10.4 getName

Returns the name of the Viewport.

Type	String
Visibility	public
IsAbstract	false
Parameter	

7.1.1.10.5 getViewEye

Returns a copy of the current ViewEyeProperties of the Viewport. Modification to the returned ViewEyeProperties object will have no effect on the Viewport. To effect change, call the setViewEye method with the desired properties.

Type	ViewEyeProperties
Visibility	public
IsAbstract	false
Parameter	

7.1.1.10.6 removeViewportChangeListener

Unregisters the givenViewportChangeListener from this Viewport. Afterwards the ViewportChangeListener will not be notified of Viewport changes. Unregistering a ViewportChangeListener that was not registered does not have any effect.

Type	void
Visibility	public
IsAbstract	false
Parameter	• in listener : ViewportChangeListener

7.1.1.10.7 scaleToPoints

Offset and scale the viewport to contain all points (as possible) passed in the given GeoPositions array. The margin Distance parameter specifies an additional space that needs to be visible around the broadest points in the points list.

Type	void
Visibility	public
IsAbstract	false
Parameter	• in margin : double (in meters) - if the margin value is negative, it is handled as 0. • in points : GeoPosition [1..n]

7.1.1.10.8 setName

Set the name of the Viewport to the given name.

Type	void
Visibility	public
IsAbstract	false
Parameter	• in name : String

7.1.1.10.9 setViewEye

Set the ViewEye properties of the Viewport to the given ViewEyeProperties. If any of the attributes of the given ViewEyeProperties object are not set, the current value of the Viewport for that attribute will not be affected.

Later changes to the given ViewEyeProperties object do not have an effect on the Viewport.

Type	void
Visibility	public
IsAbstract	false
Parameter	• in viewEyeProps : ViewEyeProperties

7.1.1.11 ClassViewportChangeEvent

ViewportChangeEvent is the type of event that is passed to a ViewportChangeListener each time the Viewport changes. Through the ViewportChangeEvent it is possible to know which Viewport has been changed and details of the change.

If there are properties other than the view eye that the client is interested in, they can query the associated Viewport Object contained in the event available in the getSource() method.

Figure 7.20 - ViewportChangeEvent

Name	ViewportChangeEvent
Qualified Name	org::omg::tacsit::controller::ViewportChangeEvent
Visibility	public
Abstract	true

7.1.1.11.1 getSource

Returns the Viewport that has changed associated with this event.

Type	void
Visibility	public
IsAbstract	false
Parameter	

7.1.1.11.2 getViewEyeProperties

Returns a copy of the ViewEyeProperties of the Viewport that has changed.

Type	ViewEyeProperties
Visibility	public
IsAbstract	false
Parameter	

7.1.1.12 Interface ViewportChangeListener

ViewportChangeListener is the type of object that is notified about a Viewport that it is registered for notification of changes.

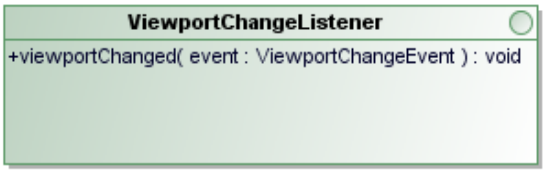


Figure 7.21 - ViewportChangeListener

Name	ViewportChangeListener
Qualified Name	org::omg::tacsit::controller::ViewportChangeListener
Visibility	public
Base Classifier	

7.1.1.12.1 viewportChanged

This method is called by the Viewport where this ViewportChangeListener is registered in case a Viewport is changed. The details of the change can be obtained through the given ViewportChangeEvent.

Type	void
Visibility	public
IsAbstract	false
Parameter	• in event : ViewportChangeEvent

7.1.1.13 Interface ViewportManager

The ViewportManager enables a TacSit client to manage Viewport instances and track changes to the set of Viewports. Construction of a Viewport is outside of the scope of this interface. Once Viewport objects are created they can be added to a manager via the addViewport method.

Figure 7.22 - ViewportManager

Name	ViewportManager
Qualified Name	org::omg::tacsit::controller::ViewportManager
Visibility	public
Base Classifier	

7.1.1.13.1 addViewport

Add a Viewport to this ViewportManager. If the Viewport is already added to the ViewportManager before, this operation has no effect.

Type	void
Visibility	public
IsAbstract	false
Parameter	• in viewport : Viewport

7.1.1.13.2 addViewportManagerListener

Registers the given ViewportManagerListener to this ViewportManager. After registration the ViewportManagerListener will be notified through each time that a viewport is added or removed through its corresponding method. Registering a ViewportManagerListener that was already registered with this ViewportManager does not have any effect.

Type	void
Visibility	public
IsAbstract	false
Parameter	• in listener : ViewportManagerListener

7.1.1.13.3 getViewport

Returns the Viewport with the given name. If there is no such Viewport, an error condition is given. (specified by the PSM)

Type	Viewport
Visibility	public
IsAbstract	false
Parameter	• in name : String

7.1.1.13.4 getViewports

Returns all Viewports managed by this ViewportManager.

Type	Viewport[]
Visibility	public
IsAbstract	false
Parameter	

7.1.1.13.5 removeViewport

Remove the given Viewport from this ViewportManager. If the given Viewport is not managed by this ViewportManager, this operation as no effect.

Type	void
Visibility	public
IsAbstract	false
Parameter	• in viewport : Viewport

7.1.1.13.6 removeViewportManagerListener

Unregisters the givenViewportManagerListener from this ViewportManager. Afterwards the ViewportManagerListener will not be notified of added or removed Viewports.

Unregistering a ViewportManagerListener that was not registered does not have any effect.

Type	void
Visibility	public
IsAbstract	false
Parameter	• in listener : ViewportManagerListener

7.1.1.14 Interface ViewportManagerEvent

ViewportManagerEvent is the type of event that is passed to a ViewportManagerListener each time a Viewport is added or removed from the ViewportManager. Through the ViewportManagerEvent it is possible to know which Viewport has been added to which ViewportManager.

Figure 7.23 - ViewportManagerEvent

Name	ViewportManagerEvent
Qualified Name	org::omg::tacsit::controller::ViewportManagerEvent
Visibility	public
Base Classifier	

7.1.1.14.1 getSource

Returns the ViewportManager to which a Viewport has been added or removed.

Type	ViewportManager
Visibility	public
IsAbstract	false
Parameter	

7.1.1.14.2 getViewport

Returns the viewport that has been added or removed by this event.

Type	Viewport
Visibility	public
IsAbstract	false
Parameter	

7.1.1.15 InterfaceViewportManagerListener

ViewportManagerListener is the type of object that is notified by a ViewportManager where it is registered in case a Viewport is added or removed from that ViewportManager.

Figure 7.24 - ViewportManagerListener

Name	ViewportManagerListener
Qualified Name	org::omg::tacsit::controller::ViewportManagerListener
Visibility	public
Base Classifier	

7.1.1.15.1 viewportAdded

This method is called by the ViewportManager where this ViewportManagerListener is registered in case a Viewport is added.

The details of the addition can be obtained through the given ViewportManagerEvent.

Type	void
Visibility	public
IsAbstract	false
Parameter	• in event : ViewportManagerEvent

7.1.1.15.2 viewportRemoved

This method is called by the ViewportManager where this ViewportManagerListener is registered in case a Viewport is removed.

The details of the removal can be obtained through the given ViewportManagerEvent.

Type	void
Visibility	public
IsAbstract	false
Parameter	• in event : ViewportManagerEvent

7.1.1.16 EnumerationSelectionMethodology

SelectionMethodology determines the scope of the SelectionManagers of the TACSIT Controller API.

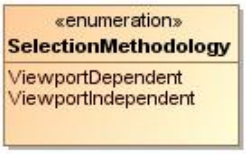


Figure 7.25 - SelectionMethodology

Name	SelectionMethodology
Qualified Name	org::omg::tacsit::controller::SelectionMethodology
Visibility	public
Base Classifier	

If the SelectionMethodology is ViewportDependent each Viewport of the TacsitController has its own SelectionManager. Therefore the selection on one viewport does not depend on the selection on another viewport.

If SelectionMethodology is ViewportIndependent all Viewports of the TacsitController have a common SelectionManager. Therefore the selection on one viewport depends on the selection on another viewport: the selections on multiple viewports are identical.

7.1.1.17 EnumerationSelectionType

SelectionType allows distinguishing different kinds of selection in the Tacsit.

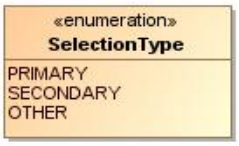


Figure 7.26 - SelectionType

Name	SelectionType
Qualified Name	org::omg::tacsit::controller::SelectionType
Visibility	public
Base Classifier	

The names of the instances are self explanatory.

7.2 Package org.omg.tacsit.query

The org.omg.tacsit.query package contains the interfaces for building queries about Entities that are used by the controller.

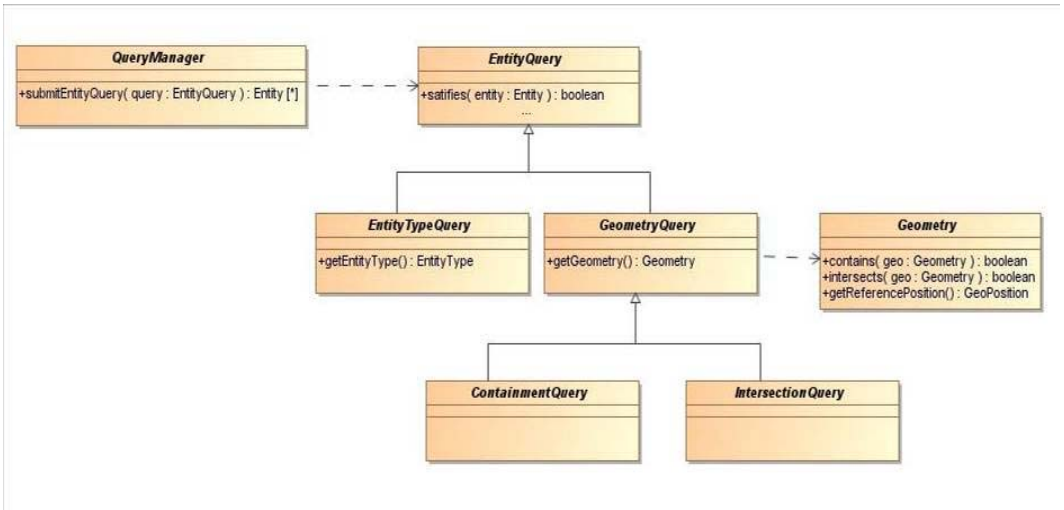


Figure 7.27 - Query Interfaces

The query interfaces are designed to provide maximum flexibility to the implementing system while still providing a useful API. The pattern for executing a query is to “build” a query using the appropriate EntityQuery objects (or its sub-classes) and to submit that query to the QueryManager. The query manager will return the Entity objects that meet the query “satisfies” method.

The basic EntityQuery is intended to provide an equals comparison on the entity object to “satisfy” the query. More specialized queries can be created by overriding the “satisfies” method. With these queries the client application can compare to any attribute in its system-specific Entity Object and query it from the TACSIT Controller.

7.2.1 Class / Interface Specifications

The following provides a detailed description of the interface classes in the query package.

7.2.1.1 Interface ContainmentQuery

The ContainmentQuery is used to determine whether or not an Entity is completely contained within a Geometry. Containment is determined by using the Entity's Geometry as an argument to the contains() method on the Geometry specified by this Query, i.e., this.satisfies(entity) = this.getGeometry().contains(entity.getGeometry())

Figure 7.28 - ContainmentQuery

Name	ContainmentQuery
Qualified Name	org::omg::tacsit::controller::ContainmentQuery
Visibility	public
Base Classifier	• GeometryQuery

7.2.1.2 Interface EntityQuery

An EntityQuery encapsulates the criteria against which an Entity may be evaluated for membership.

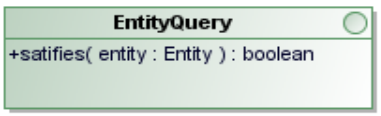


Figure 7.29 - EntityQuery

Name	EntityQuery
Qualified Name	org::omg::tacsit::query::EntityQuery
Visibility	public
Base Classifier	

7.2.1.2.1 satifies

Returns true if the specified Entity satisfies the criteria represented by this EntityQuery; false otherwise.

Type	boolean
Visibility	public
IsAbstract	false
Parameter	• in event : Entity

7.2.1.3 Interface EntityTypeQuery

The EntityTypeQuery is used to determine whether or not a given Entity belongs to a particular EntityType.

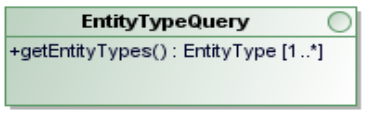


Figure 7.30 - EntityTypeQuery

Name	EntityTypeQuery
Qualified Name	org::omg::tacsit::query::EntityTypeQuery
Visibility	public
Base Classifier	• EntityQuery

7.2.1.3.1 getEntityTypes

Returns the EntityType to compare with.

Type	EntityType
Visibility	public
IsAbstract	false
Parameter	

7.2.1.4 Interface GeometryQuery

The GeometryQuery is used to determine whether or not a given Entity has a geometrical relationship with a particular Entity.

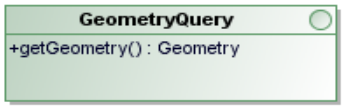


Figure 7.31 - GeometryQuery

Name	GeometryQuery
Qualified Name	org::omg::tacsit::query::GeometryQuery
Visibility	public
Base Classifier	• EntityQuery

7.2.1.4.1 getGeometry

Returns the Geometry against which an Entity will be evaluated..

Type	Geometry
Visibility	public
IsAbstract	false
Parameter	

7.2.1.5 Interface IntersectionQuery

The IntersectionQuery is used to determine if an Entity intersects geometrically with a Geometry. Intersection is determined by using the Entity's Geometry as an argument to the intersects() method on the Geometry specified by this Query, i.e., this.satisfies(entity) = this.getGeometry().intersects(entity.getGeometry())

Figure 7.32 - IntersectionQuery

Name	IntersectionQuery
Qualified Name	org::omg::tacsit::query::IntersectionQuery
Visibility	public
Base Classifier	• GeometryQuery

7.2.1.6 InterfaceQueryManager

The QueryManager supports the execution of queries for entities against the TACSIT Controller.

Queries submitted through the QueryManager are transient in nature; that is, the results of a query represent the state of the TacSit Controller at the time the Query was executed. Query results will not be maintained by the QueryManager as the state of the TacSit Controller changes. Since the QueryManager treats the Query objects as stateless, the same Query may be re-used for any number of subsequent submissions.

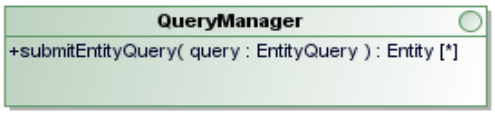


Figure 7.33 - QueryManager

Name	QueryManager
Qualified Name	org::omg::tacsit::query::QueryManager
Visibility	public
Base Classifier	

7.2.1.6.1 submitEntityQuery

Returns the set of all Entities that satisfy the criteria expressed in the given EntityQuery..

Type	Entity
Visibility	public
IsAbstract	false
Parameter	• in query : EntityQuery

7.3 Package org.omg.tacsit.geometry

The org.omg.tacsit.geometry package contains the interfaces for describing the location and bounds of regions of interest to the TACSIT Controller API.

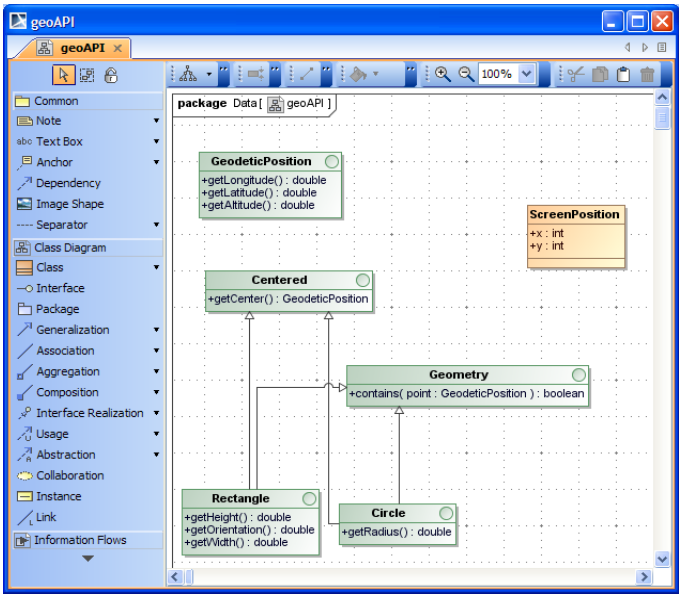


Figure 7.34 - Query Interfaces

The interfaces described here are the minimal subset of possible geometries for interating with the TACSIT.

7.3.1 Class / Interface Specifications

The following provides a detailed description of the interface classes in the query package.

7.3.1.1 Interface Geometry

Geometry is the base class for all representations of geometric shapes. Every Geometry must be able to evaluate containment and intersection criteria between itself and any other Geometry. For non-closed Geometries, or for Geometries with edges that cross, it is up to those Geometries to define containment of a specified point within that Geometry.

This OMG specification is consistent with the OGC “OpenGIS® Implementation Specification for Geographic information - Simple feature access - Part 1: Common architecture” (Doc: OGC 06-103r3; Version: 1.2.0; Date: 2006-10-05) in that this OMG specification does not contradict the OGC specification and uses the same names for types and methods where possible.

This OMG Specification does not require a Geometry to completely implement the OGC specification.

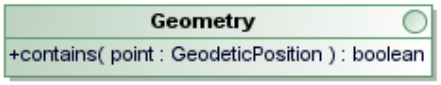


Figure 7.35 - Geometry

Name	Geometry
Qualified Name	org::omg::tacsit::geometry::Geometry
Visibility	public
Base Classifier	• Cloneable • Serializable

7.3.1.1.1 contains

Returns TRUE if the passed GeodeticPosition object “point” projected on the surface of the Earth is contained within this Geometry object. .

Type	boolean
Visibility	public
IsAbstract	false
Parameter	• in geo : GeodeticPoint

7.3.1.2 Interface GeodeticPosition

GeoPosition represent a geographical location i.e., a position on the earth.

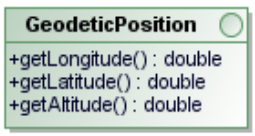


Figure 7.36 - GeodeticPosition

Name	GeodeticPosition
Qualified Name	org::omg::tacsit::geometry::GeodeticPosition
Visibility	public
Base Classifier	• Geometry

7.3.1.2.1 getLatitude

Returns the latitude in Radians.

Type	double
Visibility	public
IsAbstract	true
Parameter	

7.3.1.2.2 getLongitidue

Returns the longitude in Radians.

Type	double
Visibility	public
IsAbstract	true
Parameter	none

7.3.1.2.3 getAltitude

Returns the altitude in meters or {Double.NAN} if the altitude is not valid.

Type	double
Visibility	public
IsAbstract	true
Parameter	none

7.3.1.3 Class ScreenPosition

ScreenPosition is a simple class to encapsulate a point location on the screen.

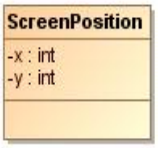


Figure 7.37 - ScreenPosition

Name	ScreenPosition
Qualified Name	org.omg.tacsit.geometry.coordinate::ScreenPosition
Visibility	public
Abstract	false

7.3.1.3.1 x

The x position of the point.

Type	int
Default Value	
Visibility	public
Multiplicity	1

7.3.1.3.2 y

The y position of the point.

Type	int
Default Value	
Visibility	public
Multiplicity	1

7.3.1.4 Interface Centered

Centered is a simple class to designate that a particular Geometry object is centred at a certain GeodeticPosition.

Figure 7.38 - Centered

Name	Centered
Qualified Name	org::omg::tacsit::geometry::Centered
Visibility	public
Base Classifier	

7.3.1.4.1 getCenter()

Returns the center of the geometry..

Type	GeodeticPosition
Visibility	public
IsAbstract	true
Parameter	none

7.3.1.5 Interface Circle

Circle is a simple centred circular geometry with a radius measured in meters.

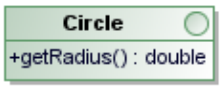


Figure 7.39 - Circle

Name	Circle
Qualified Name	org::omg::geometry::Circle
Visibility	public
Base Classifier	• Centered • Geometry

7.3.1.5.1 getRadius()

Returns the radius of the circle in meters.

Type	double
Visibility	public
IsAbstract	false
Parameter	

7.3.1.6 Rectangle

Rectangle is a simple centred geometry with a width and height measured in meters. An orientation is provided to skew the rectangle as necessary.

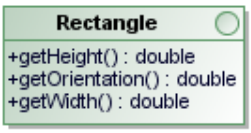


Figure 7.40 - Rectangle

Name	Rectangle
Qualified Name	org::omg::tacsit::geometry::REctangle
Visibility	public
Base Classifier	• Centered • Geometry

7.3.1.6.1 getHeight

Returns the height of the rectangular shape in meters.

Type	double
Visibility	public
IsAbstract	false
Parameter	

7.3.1.6.2 getOrientation

Returns the azimuth of the y axis of the rectangle shape, where zero is north in radians.

Type	double
Visibility	public
IsAbstract	false
Parameter	

7.3.1.6.3 getWidth

Returns the width of the rectangle shape in meters.

Type	double
Visibility	public
IsAbstract	false
Parameter	

8 Java Platform Specific Model

A Java PSM is provided separately. The PSM contains a Java-specialized UML Model, and sample Java classfiles of the requisite classes.

Note that in the Java PSM, the ScreenPosition class is replaced by the standard Java representation of a screen position (java.awt.Point). This is noted in the Java UML Model.

Additionally, instead of using array types for parameters and return types with multiple values, a templated version of the standard Java Collection Object (List) java.util.List<NecessaryType> has been used.

9 C++ Platform Specific Model

A C++ PSM is provided separately. The PSM contains a sample C++ header files of the requisite classes.